05

Interfaces

Transcripción

[00:00] Y bien, ¿se les ocurrió alguna otra solución por el momento? Esto seguro que sí, y hay muchas. La creatividad en programación es lo principal. Vamos a implementar esto del código, vamos a crear nuestro cliente. Mejor dicho el cliente ya está aquí, existe, y vamos a renombrar esta clase autenticable, que no me está llamando con otra función autenticable. Ya hemos quedado que la vamos a renombrar a autenticable. Perfecto.

[00:31] Entonces solamente le cambiamos el nombre aquí y nos está dando error. Y esto es una cosa más que me gustaría que presten atención. En el caso de Java el nombre de la clase pública tiene que ser igual al nombre del archivo. Por ejemplo, si ustedes ven las sugerencias que me están saliendo ahí abajo, sale, bueno, ¿el error qué te dice?

[00:58] El nombre público autenticable debe ser definido en el propio archivo. ¿Qué quiere decir eso? Ya que yo le cambié el nombre autenticable, el nombre del archivo aquí debería ser también autenticable. ¿Qué sugerencias me da Eclypse? Renombrar el archivo a autenticable.java o renombrar la clase a FuncionarioAutenticable.

[01:20] Yo voy a elegir la primera y él lo va a hacer por mí automáticamente y listo. Y vemos que saltó error en administrador y en gerente. ¿Por qué? Porque ya no existe pues autenticable, ¿no? Entonces vamos a arreglar esos pequeños errores que están saliendo aquí. Guardamos acá y de igual forma con el gerente. Perfecto, y listo. Volvió todo a la normalidad.

[01:46] Y aquí también como parámetro lo estábamos mandando. Listo. Y volvió todo a la normalidad, como les prometí. Bien. Ahora que tenemos todo a la normalidad volvemos a nuestro cliente y cerramos todo lo demás, para mantener el orden. Y nuestro cliente también se va a volver autenticable. Perfecto.

[02:21] Y listo, nuestro cliente ya es un cliente autenticable. Entonces aparte de tener su nombre, documento, teléfono, ahora él también puede tener su contraseña y acceso al sistema, perfecto. Entonces, ya habíamos quedado que esa solución no era del todo práctica. Recapitulando, no era del todo práctica. ¿Por qué?

[02:46] Porque autenticable si bien es un nombre que es neutral, que se puede aplicar para cliente, administrador y gerente, la abstracción de autenticación en sí ¿está a cargo de quién? Del sistema interno. Él es el encargado de hacer que todos accesen al sistema. Él va a ser nuestra única puerta de entrada. Y para eso el facilitador, ¿quién va a ser? Autenticable. Él va a dejar entrar a todos aquellos que le digan: "Oye, por si acaso yo soy autenticable".

[03:19] ¿Entienden? Entonces él es la puerta y autenticable es la llave por así decirlo. Todos los que tengan autenticable en su carnet por ejemplo ya está, el mejor ejemplo que se me puede ocurrir. Autenticable es como el carnet que tú usas para entrar a tu trabajo, ese sensor que usas para marcar asistencia, ese es digamos tu autenticable, ese debe ser tu extend de autenticable. Tú puedes entrar a tu trabajo solo si tienes tu carnet. Listo.

[03:50] O a tu centro de estudios también, ¿no? Solo te dejan entrar si tú muestras tu carnet, muestras pues que eres parte de ese sistema, de esa casa de estudios y entras. Listo. Pero el error conceptual entra aquí, en esta relación, que es lo que no me está gustando para nada, y es que el autenticable es funcionario.

[04:11] Sería muy bueno, sería excelente que por ejemplo autenticable está aquí, vamos a dejar esto aquí. Sería muy bueno que autenticable esté aquí y que a su vez podamos separar digamos esa funcionalidad digamos o ese concepto de que el gerente también pueda ser funcionario. ¿Cierto? Si tuviera esta habilidad, digamos, si tuviera este enlace de poder extender de los dos, de poder ser hijo de los dos, sería excelente pero no lo tiene.

[04:57] No lo tiene a nivel de herencia pero lo tiene a nivel de polimorfismo. ¿Qué quiere decir eso? Tenemos una salida que es las interfaces. Ustedes se preguntarán: "¿Interfaces? ¿Qué es una interfaz?" Conceptualmente una interfaz vendría a ser algo parecido a lo que ya conocemos como clase. Una clase es así: define atributos, define método, tiene un nombre.

[05:26] ¿Pero cuál es la diferencia a la interfaz? Es muy parecido a lo que puede ser nuestra clase abstracta. Por ejemplo, vamos a autenticable y ya hemos visto, hay un punto muy bueno aquí y es que si la clase también es abstracta, no necesita implementar los métodos de la clase padre, ¿cierto? Entonces, si yo hiciera esto de aquí, abstract class, y yo borro este de acá, me da error en cliente.

[06:10] ¿Por qué? Porque cliente no está implementando el método getBonificacion. Bueno, esto sería otro error causado pues por este modelo que estoy usando. Aquí ya vimos un caso que yo no tenía mapeado, pero bueno, es un error que también vamos a tener, y es que el cliente tendría un método que no le corresponde para nada. Pero bueno, vamos a dejar eso de lado.

[06:30] Ya sabemos que está mal, vamos a enfocarnos en la solución. Vamos a volver a autenticable, que en este caso sería pues abstracto también y al ser abstracto ya hemos quedado pues que él define sus propios métodos, él define la firma del método pero puede dejarlo sin cuerpo para que cada uno lo implemente a su manera. De repente la clave en el caso de los administradores es estrictamente numérica y en el caso de los gerentes puede ser alfanumérica.

[07:10] Es un ejemplo que se me está viniendo a la cabeza. Pero hay algo más, hay algo más aquí, vamos a probar por ejemplo borrar primero este cuerpo de acá, y lo dejo así. Perfecto. Y ya sabemos que para que compile una palabrita reservada más: abstract. Entonces, él puede setear la clave, ¿pero cómo inicia sesión? Problema tuyo. Llega aquí, nos va a dar el error. ¿Por qué?

[07:40] Porque nos está diciendo: "necesitas implementar el método iniciarSesion". Por defecto me va a retornar false, que es el valor por defecto de todo boolean, y yo aquí debería implementar mi lógica de negocio. De igual forma en el administrador. Para cada clase que extienda de Autenticable, yo debería de implementar mi método iniciarSesion.

[08:10] Pero aún así, aunque también sea una clase abstracta, no me deja cumplir con este modelo que yo estoy queriendo implementar y que de hecho es el que más me conviene, que a la vez el gerente sea autenticable y sea funcionario, porque si el gerente es autenticable, por autenticable nosotros definimos cualquier objeto que inicie sesión en el sistema interno.

[08:35] Esto está completamente desacoplado si es del funcionario o no. Y aquí entra lo que les mencioné de las interfaces. ¿Por qué? Porque es muy parecido a una clase abstracta. Vamos a ver acá, solamente que vamos a ponerle, en vez de clase, interface. Perfecto. Y la interfaz no puede tener métodos implementados.

[09:06] Ese es el primer punto que van a ver aquí que es el setClave. Si yo dejara esto así, perfecto, compila. Pero la interfaz no puede tener métodos ya implementados, no puede tener métodos con cuerpo, ese es el primer punto de la interfaz. Ahora, en Java8 ya se implementó lo que es la interfaz default, que es un concepto en sí un poco más avanzado.

[09:35] Pero bueno, una interfaz permite tener un solo método implementado, olvídenlo. Esto es solamente digamos para que lo tengan presente, pero no es tan utilizado que digamos y de hecho que lo vamos a ver en los cursos ya más

avanzados de Java. Esto es solamente un spoiler de lo que se puede venir más adelante, pero vamos a dejarlo así por el momento.

[09:58] Por ahora lo importante es entender que la interfaz no tiene métodos implementados.